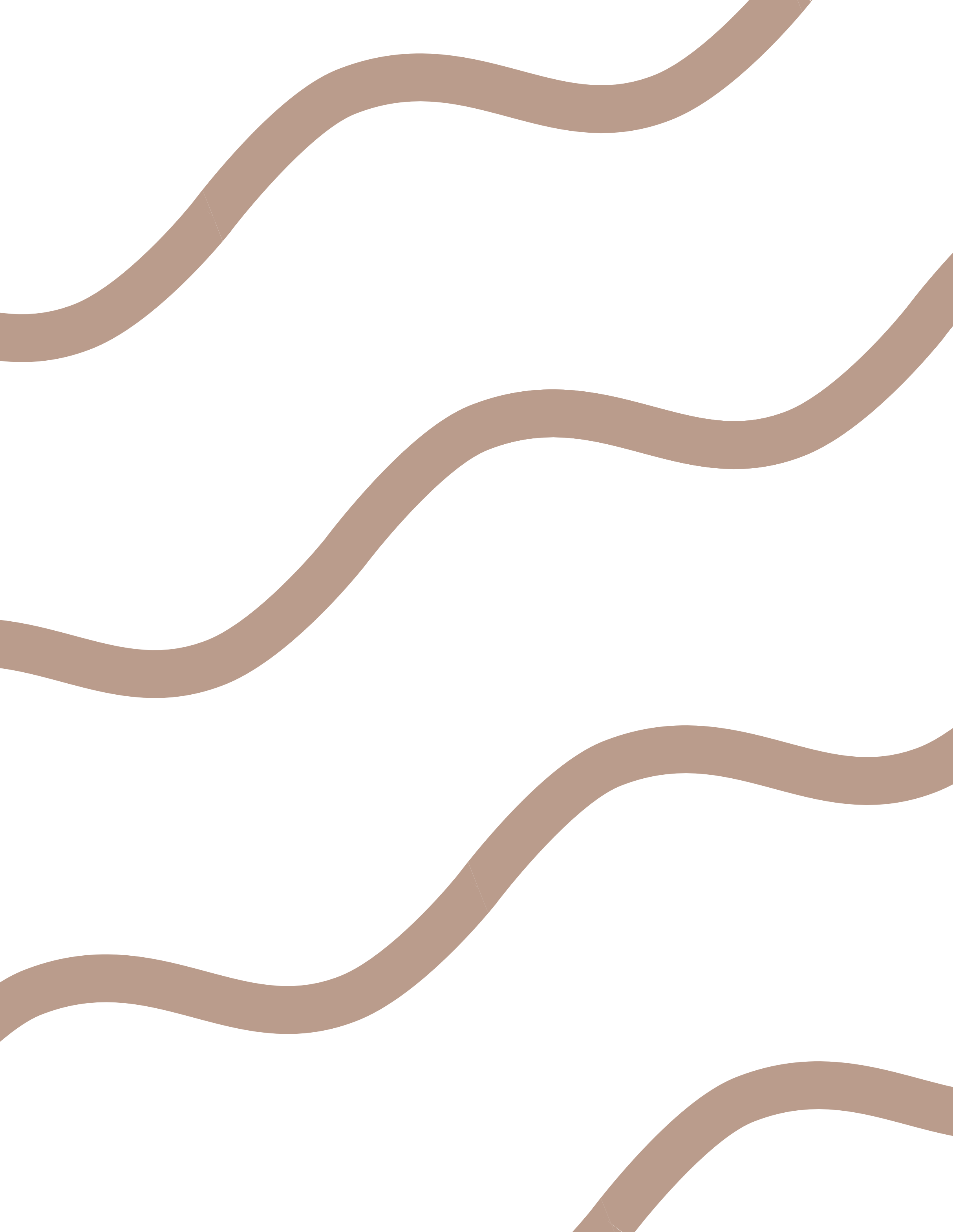
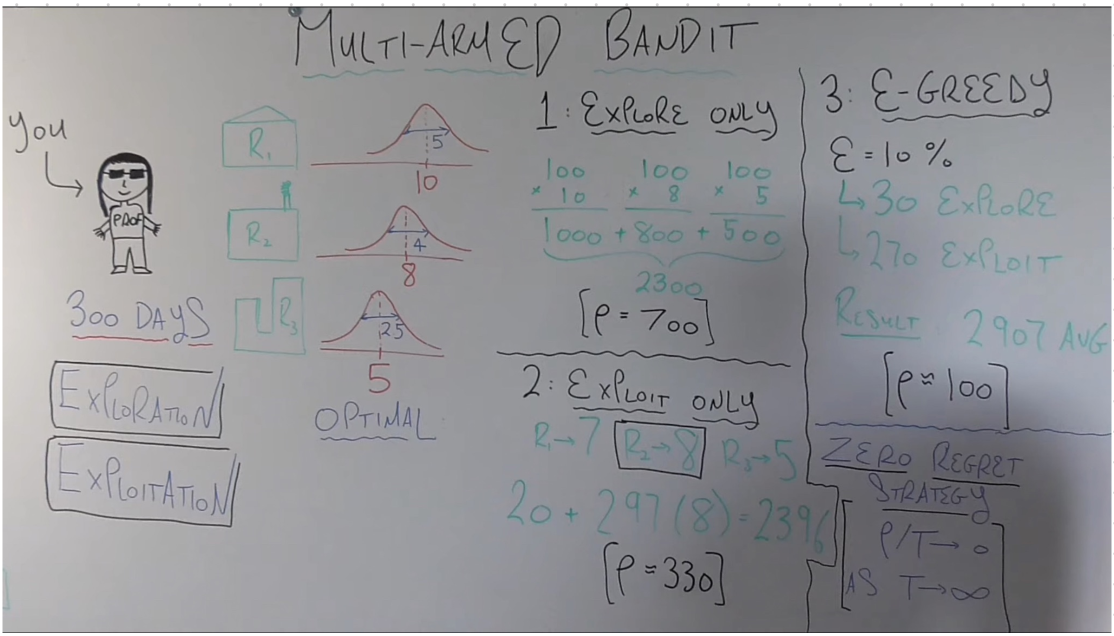


Fundamentals of Reinforcement Learning



ρ → <u>regret</u> → Diff. b/w max happiness available − happiness from your strategy

'rho'

$q_*(a) = E[R_t \mid A_t = a]$

value of action

action selected on time step t

$Q_t(a) \rightarrow$ estimated value of action a at time step t

$$\boxed{Q_{n+1} = Q_n + \alpha[R_n - Q_n]}$$

Increment Update rule

New estimate = Old estimate + stepsize [Target − Old Estimate]

Q_n for nonstationary Problem

$$\boxed{Q_{n+1} = (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i}$$

$\Rightarrow A_t \rightarrow$ action at time t

Ways of balancing Exploration & Exploitation

I Epsilon greedy

exploitation

$$A_t \leftarrow \begin{cases} \arg\max_a Q_t(a) & \text{with probabily } 1-\varepsilon \\ a \sim \text{Uniform}(\{a_1 \ldots\ldots a_k\}) & \text{with probability } \varepsilon \end{cases}$$

exploration (random)

II. Optimistic initial value

III

Upper-Confidence Bound (UCB) Action Selection

uncertainity term

$$A_t = \arg\max \left[\underbrace{Q_t(a)}_{\text{exploit}} + \underbrace{c\sqrt{\frac{\ln t}{N_t(a)}}}_{\text{explore}} \right]$$

$$c\sqrt{\frac{\ln t}{N_t(a)}} \longrightarrow c\sqrt{\frac{\ln \text{timesteps}}{\text{times action a taken}}}$$

$$c\sqrt{\frac{\ln 10000}{5000}} \longrightarrow 0.043c$$

$$c\sqrt{\frac{\ln 10000}{100}} \longrightarrow 0.303c$$

less time → more uncertainity

IV Gradient Bandit

X X X X

Markov Decision Process

$$p(s', r \mid s, a) = Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$

$S_t \in S$ → set of possible states

$A_t \in A(S_t)$ → set of possible actions at that state

$$\sum_{s' \in S} \sum_{r \in R} p(s', r \mid s, a) = 1 \quad \text{for all } s \in S, a \in A(s)$$

$$r(s, a) = E[R_t \mid S_{t-1} = s, A_{t-1} = a] =$$

$$r(s, a, s') = E[R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s]$$

$0 \leq \gamma \leq 1 \Longrightarrow$ Discount rate

Return at timestep t (episodic tasks)

$$G_t = R_{t+1} + R_{t+2} + \ldots\ldots + R_T \longrightarrow \text{terminal state}$$

Contious Tasks

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots\ldots + \gamma^{k-1} R_{t+k} + \ldots$$

$$= \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

$$\boxed{G_t = R_{t+1} + \gamma G_{t+1}} \longrightarrow \text{recursive defination}$$

If γ approaches $1 \longrightarrow$ long term
(take future rewards into more consideration)

$$\boxed{G_t = \sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma}}$$

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \leq \sum_{k=0}^{\infty} \gamma^k R_{max} = R_{max} \sum_{k=0}^{\infty} \gamma^k$$

constant

$$= \boxed{R_{max} \times \frac{1}{1-\gamma}}$$

(it's not changing)
(every action gets R_{max})

Policies

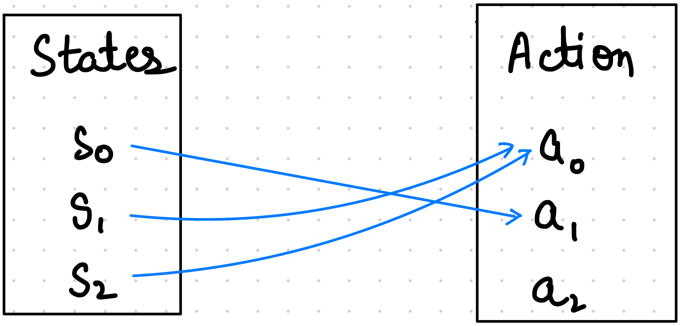
$\longrightarrow$ specifies how an agent behaves

Policy is a distribution over actions for each possible state.

$\longrightarrow$ A policy maps each state to a single action.

$$\boxed{\pi(s) = a}$$

$\longrightarrow$ represents the action selected in state 's' by policy 'π'.



Deterministic policy notation

In general policy assign probabilities to each action in each state.

$\pi(a|s) \longrightarrow$ probability of selecting action a in a state s.

$$\sum_{a \in A(s)} \pi(a|s) = 1 \quad ; \quad \pi(a|s) \geq 0$$

Stochastic policy notation

Value Functions

$R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3}$

I State Value Functions

$$v_\pi(s) \doteq E_\pi[G_t \mid S_t = s] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right]$$

a state value function is the future award an agent can expect to recieve starting from a particular state.

- (expected return from a given state)

II Action - value Function

$G_t = R_t + \gamma G_{t+1}$

$$q_\pi(s,a) \doteq E_\pi[G_t \mid S_t = s, A_t = a]$$

an action value describes what happens when the agent first selects a particular action.

- (the action value of a state is the expected return if the agent selects action A and then follows policy 'π')

State value functions represent the expected return from a given state under a specific policy

Action-value functions represent the expected return from a given state after taking a specific action, later following a specific policy

In reinforcement learning, "value functions" are a fundamental concept used to estimate and evaluate the desirability of different states and state-action pairs within an environment. Value functions play a crucial role in guiding an agent's decision-making process and learning how to maximize cumulative rewards. There are two main types of value functions:

1. **State-Value Function (V-function, V):**
   - The state-value function, denoted as V(s), estimates the expected return or cumulative reward an agent can achieve starting from a specific state while following a particular policy.
   - V(s) quantifies how good it is to be in a particular state under a given policy. It represents the long-term expected reward the agent can expect if it starts in state s and continues to interact with the environment according to the policy.
   - The formula for the state-value function is typically expressed as:
     $V(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | S_0 = s\right]$
     where E denotes the expectation, γ is a discount factor that gives less weight to distant rewards, R_{t+1} is the reward received at time step $t+1$, and S_0 is the initial state.
2. **Action-Value Function (Q-function, Q):**
   - The action-value function, denoted as Q(s, a), estimates the expected return or cumulative reward an agent can achieve when it starts from a specific state, takes a particular action, and then continues to interact with the environment following a specific policy.
   - Q(s, a) quantifies the desirability of taking a particular action, a, in a specific state, s, under the given policy.
   - The formula for the action-value function is often expressed as:
     $Q(s, a) = E\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | S_0 = s, A_0 = a\right]$
     where A_0 is the initial action taken.

Regenerate

Value functions provide the agent with a way to assess and compare different states or state-action pairs in terms of their expected long-term rewards. The goal of reinforcement learning is to find the optimal policy (the best strategy) that leads to the highest values in states, and, in turn, maximizes cumulative rewards. Various algorithms, such as dynamic programming, Monte Carlo methods, and temporal difference learning, are used to estimate and update value functions to achieve this goal.

Bellman Equation

State-value Bellman equation

Bellman equation for a state value function defines a relationship between the value of a state and the value of his possible successor states.

$$v_\pi(s) \doteq E_\pi[G_t \mid S_t = s]$$

$$= E_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s]$$

$$= \sum_a \pi(a,s) \sum_{s'} \sum_r p(s', r \mid s, a)\left[r + \gamma E_\pi[G_{t+1} \mid S_{t+1} = s']\right]$$

$$= \sum_a \pi(a,s) \sum_{s'} \sum_r p(s', r \mid s, a)\left[r + \gamma v_\pi(s')\right]$$

Action-value Bellman equation

$$q_\pi(s,a) = E_\pi[G_t \mid S_t = s, A_t = a]$$

action is already fixed

$$= \sum_{s'} \sum_r p(s', r \mid s, a)\left[r + \gamma E_\pi[G_{t+1} \mid S_{t+1} = s']\right]$$

$$= \sum_{s'} \sum_r p(s', r \mid s, a)\left[r + \gamma \sum_{a'} \pi(a' \mid s') E_\pi[G_{t+1} \mid S_{t+1} = s', A_{t+1} = a']\right]$$

$$= \sum_{s'} \sum_r p(s', r \mid s, a)\left[r + \gamma \sum_{a'} \pi(a' \mid s') q_\pi(s', a')\right]$$

Optimal Policy

An optimal policy π_* is a policy which is as good as or better than all the other policies.

$\Longrightarrow$ $\pi_1 \geq \pi_2$ if & only if $v_{\pi_1}(s) \geq v_{\pi_2}(s)$ for all $s \in S$.

Optimal value function

v_* $v_{\pi_*}(s) \doteq \mathbb{E}_{\pi_*}[G_t | S_t = s] = \max_{\pi} v_\pi(s)$ for all $s \in S$

q_* $q_{\pi_*}(s,a) = \max_{\pi} q_\pi(s,a)$ for all $s \in S$ and $a \in A$

$$v_\pi(s) = \sum_a \pi(a,s) \sum_{s'} \sum_r p(s', r | s,a) [r + \gamma v_\pi(s')]$$

(bellman eq.)

$$v_*(s) = \sum_a \pi_*(a,s) \sum_{s'} \sum_r p(s', r | s,a) [r + \gamma v_*(s')]$$

$$v_*(s) = \max_a \sum_{s'} \sum_r p(s', r | s,a) [r + \gamma v_*(s')]$$

Bellman Optimality Equation for v_*

$$q_*(s,a) = \sum_{s'} \sum_{r} p(s', r \mid s, a) \left[r + \gamma \max_{a'} q_*(s', a') \right]$$

Bellman Optimality Equation for q_*

$$v_*(s) = \max_a \sum_{s'} \sum_r p(s', r \mid s, a)[r + \gamma v_*(s')]$$

$$\pi_*(s) = \arg\max_a \boxed{\sum_{s'} \sum_r p(s', r \mid s, a)[r + \gamma v_*(s')]}$$

$$r + \gamma v_*(s')$$

$$\pi_*(s) = \arg\max_a q_*(s, a)$$

DYNAMIC PROGRAMMING

(week 4)

$$[r + \gamma \overline{v_\pi(s')}^{\,0}]$$

$$\boxed{-1}$$

$$\boxed{x = \frac{3}{4}(1-x) + \frac{1}{4}(-21)}$$

$$-1 + \gamma v_\pi(s')$$

$$[-1 + (-20]$$

$$-1 + [- \quad x_2 = \frac{1}{4} \qquad x = 0.25(-21)$$

$$x_2$$

$$x = \frac{1}{4}[r + \gamma v(s')]$$

$$= \frac{1}{4}[-1 + x]$$

$$= 0.25 + 0.25x$$